1991 HONOURS PROJECT
GEOFF THOMAS

IMPROVING THE SOUNDEX ALGORITHM

IN CONJUNCTION WITH
PAXUS HEALTH LTD.

1991 Honours Project Geoff Thomas

Names matching with errors : Improving the Soundex Algorithm

ABSTRACT

Searching a database of people's names is prone to problems from many sources. Some problems are caused by operator errors on the part of the person querying the database, such as using an incorrect spelling for a name that is given over the telephone (e.g. "McKay" instead of "MacKay"). Another problem is data entry errors at the time of entry into the computer, such as accidentally transposed characters. These problems decrease the effectiveness of searching the data, and so techniques need to be introduced to give reasonable results to queries even in the presence of multiple differences between data and search key. This project outlines some of the points that need to be considered when choosing an algorithm to perform approximate matching. Some of the methods available are described and evaluated, and the results are presented.

1. INTRODUCTION

Mr. John Van der Meer is brought into hospital one night with a fractured skull, after drinking and driving, and colliding with a lamp-post. He has no identifying documents on him, except for his student identification card. The doctor wishes to find out if they have Mr. Van der Meer on their records, so she gives the computer operator at the front desk the ID card, and asks him to locate Mr. Van der Meer's record.

The operator enters "Van der Meer" into the computer, and presently it responds with a statement that there is no surname "Van der Meer" in the database. It then presents a list of related surnames of people that have visited the hospital recently, including a group of four surnames of "Vander Meer". The operator searches through this list of people, and notices that one of them is dead, one has a different christian name, and the third is only three months old. The fourth name is a "John Vander Meer" that fits the description of the patient, and the operator notes that this person is a diabetic with an allergy to penicillin.

It is easy to see the outcome of the operator misinforming the doctor as to Mr. Van der Meer's state of health, as a result of not being able to gain access to his record.

This honours project has been done in cooperation with the Paxus Health organization, who maintain a database of patients from hospitals around New Zealand. Currently they are using the well known Soundex algorithm for querying their database. This algorithm, while being useful in some cases, such as English surnames, has deficiencies which cause it to give poor results on many searches. For example, Polynesian names are not handled well by the Soundex algorithm, as it was not based on these. The aim of this project then, is to determine a better algorithm (or group of algorithms) to use in its place.

Retrieving records from a database, using a person's name as the search key is a very common operation. One problem that is faced is the vast range of possible, valid names using even a small character set, and another is that either the data, search key or both may contain errors or uncertain characters which limit the effectiveness of a search.

Exact matching can only determine the existence of a record in the database that is specified correctly in the query, and was correctly entered into the database originally. If there are some differences between the surname used as a search key and those retrieved from the database then the effectiveness of exact

searching is effectively useless, as it will not locate the desired record. Such differences can manifest themselves in a number of ways.

The problems encountered when searching for names vary slightly from those in searching text documents for spelling errors the range of valid names available is essentially unlimited, while the average vocabulary of a typical fluent English speaker maybe in the region of about 5000 - 10000 words. Also the exposure of an English operator to words in this vocabulary is much higher than their exposure to different surnames, and the range of variation is much lower. Even more extreme is the case of the Pascal programming language, with about 35 possible keywords that are valid.

For example, the Pascal code to display the words "Hello, world!" would contain about 6 of the available keywords from the language, and an English text might contain about 5500 distinct words (Peterson, 1980).

Because both Pascal and English have a relatively low vocabulary compared with surnames, in the sense that there are a finite and reasonably small number of valid words, 'spelling checkers' for these languages can utilize a dictionary look-up table, to ascertain the correctness of a word's spelling (Peterson, 1980). If the word does not exist in the dictionary, it is probably in error, and the user is prompted with some type of request. Such a look-up procedure is not possible with names, as there are few rules, if any, governing the correctness of a name. In the case of Pascal or English, good (or reasonably reliable) techniques are available for the detection of errors, and for suggestions of correct matches to be presented to the operator.

However, in searching for surnames the operator can only make use of suggestions of possible matches - none are known to be in error in relation to the search key, and all surnames in the database need to be considered as being possible matches, since all are potentially valid.

There are several reasons why the search key may not match the appropriate record in the database. Misspellings and pronunciation could be a major cause of differences, especially for names from a different ethnic origin, such as "Wirth" - the surname is pronounced "Veert" by a European speaker. New Zealanders have a different concept of how a name of a certain sound should be spelt, from that of say, a native French speaker. So when a New Zealander queries the database with a particular spoken French name such as "Renaud" (pronounced 'Reno'), the response would probably be different from that of a French operator. Ideally this effect should be reduced as much as possible. Another aspect of the Paxus database is that New Zealand is a very multicultural society, with many nations represented. Searching algorithms should ideally be able to handle surnames from a wide variety of different ethnic origins with the same quality of results, but in reality this is difficult, as the algorithms usually rely on features of some ethnic group to give enhanced performance for that group.

Typographical errors such as transposition, insertion, deletion and substitution are a major cause of errors in English text (Peterson, 1980). These are usually operator errors where a typing mistake has been made during entry of the query, data or both, and are generally randomly distributed over the four types of errors mentioned above.

Lack of information, when the operator only has enough information to form a partial or incomplete search key, is another possible source of errors. A case of this is if the computer operator in the example at the beginning of this text could not determine a surname, due to it being illegibly written. A related example of this is the searching of a police database for license plate on a car, where only some characters are known for certain, and others are not known, or possibly incorrect.

An important question in matching surnames is where does the information in a name exist? Prefixed names such as "MacPherson" or "Van der Boorn" contain less information in the prefix than in the remainder of the name, as this prefix is common to a large number of names.

The database used in the study is actual data, and as such contains some interesting features. There are over 130 thousand uniquely different surnames stored with the number of people with that surname, totalling over three million people who have had some contact with the health system in New Zealand. Sixty thousand of these surnames occur just once in the main database (see figure 1). Many of these are unique surnames, that are uncommon, but some appear to be erroneous versions of more common surnames, such as 'Smith'. Any method chosen should be able to generalize these single occurrences to nearby, more common names, as it would be desirable to have these names returned appropriately with more common, and therefore more statistically probable matches.

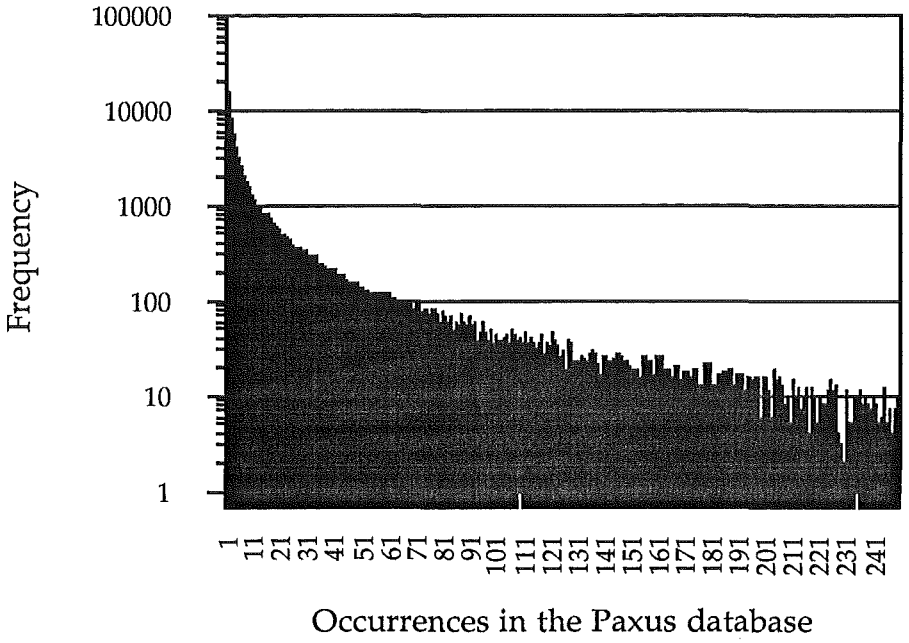


Figure 1.
Occurrences of surnames in the database

The speed requirements for online searching are strict. Time taken for a search should be short, to give good response times for a system that may be under heavy load, and Paxus Health is aiming for 15 surnames presented to the operator in less than 8 seconds. Some types of algorithm can present possibly matching surnames immediately as they are found. Others need to scan through the entire database to determine the ranking of possible matches, which gives an initial pause, but then produces a list of the best matches found for the whole database. There is also a tradeoff between the response time and the quality of results from a search; i.e. a good search algorithm that gives appropriate results in a reasonable amount of time may be much more use than a bad algorithm that gives fast but poor results.

Approximate surname matching algorithms can be divided into two types: *classification algorithms* and *distance algorithms*. A *classification* algorithm assigns each surname to a particular class based on some of its features. These classes can be determined in advance, and stored with the surnames to give fast execution of a search: the class of the search key is calculated once, and this is matched with the stored classifications for each name. *Distance* algorithms provide a metric of the similarity between two surnames. A search key is compared with every name in the database, and those with the best measure of similarity are presented in order of highest similarity. They signal a match if two words passed to them are above a certain threshold of similarity.

Another difference between these two types of algorithms is that classification based algorithms will return matches if they are in the same class, regardless of how closely they resemble the search key, and a badly chosen classification scheme will predictably produce poor results, whereas a distance based algorithm tries to find the nearest matches to the search key, giving a natural rating of

closeness, under all situations. Consequently it is hard to find a case where the distance algorithm performs badly.

These types of algorithms could be combined to form a hybrid algorithm, that roughly classified the database according to some general rules, such as 29 classes, sorted in ascending order on the first character of the surname, and then use a distance algorithm within the class. The benefits of this approach are two-fold; the speed of accessing classifications, and the accuracy of results from the search. Such an approach has not been investigated in this report, but would be worth further research in the future.

This is similar to the "calculate vs. store" tradeoff, where algorithms should be chosen according to the cost of some computing resource. If the cost of storage is high, compared with calculation, then values required should be calculated "on the fly". If however, the cost of processing is high compared with the cost of storage, then values required later should be stored, if possible.

The remainder of this report describes the Soundex algorithm, which is currently used by Paxus, and describes its merits and negative aspects relative to their database. It then presents other approximate searching methods, and an evaluation of each of these methods against Soundex concludes the report.

2. Russell's 'Soundex' algorithm

Russel's Soundex algorithm was developed in the early 1900's (Knuth, 1973) and so is quite an old algorithm by today's standards. A related method was reportedly in use in the late 1800's for processing American census data (Knuth, 1973).

It is a classification algorithm that classifies a word based on features of the word, such as particular consonants following each other, and as a result each word is placed into one of about 8900 classes. A class is chosen by selecting the first four non-vowels[1] in a surname, (for example, MacPherson is changed MCPR) and then assigning a digit to the last three according to the translations in table 1 (MCPR is translated to M216). This code can be stored in a separate field in each record of the database.

Letter:	Digit:
b, f, p, v	1
c, g, j, k, q, s, x, z	2
d, t	3
l	4
m, n	5
r	6

Table 1.
Translation table for the Soundex algorithm

This algorithm acheives a crude form of generalization of a surname by removing some of the characters, combining some others, and limiting the length of the code to a fixed value. Using this strategy it is able to skip over slight differences in most words. Because each surname translates to just one code, it can be generated automatically at data entry time, and stored within the record.

The Soundex algorithm has a number of problems associated with it. Its generalization strategy is not very good, as it is only based on individual characters, and doesn't take their context into account. For example, the consonant pair "ph" is generally pronounced "f" in most words, but Soundex does not have provisions for this type of syllabic information. Since the code is generated from the first letter and the next three consonants (if they exist), prefixes of surnames such as 'Mac' or 'Van der' take up much of the available code before the important information in the surname is reached.

The average length of surnames in the Paxus database is seven characters (figure 2). The algorithm does not use the entire surname for about 40 percent of the

[1] "Non-vowel" characters in the discussion are: all alphabetic characters used in the database, with the exception of vowels and the characters 'H', 'W' and 'Y', which are considered equivalent to vowels, and consequently discarded.

surnames from the database (figure 3). Some of these are extreme cases, where less than half of the surname is used. As a result, the number of distinct surnames in these classes is high. Another related problem is that of removing too much information by taking away the vowels. A lot of information is lost in the case of Polynesian and Asian surnames, which have a large proportion of vowels in relation to consonants. For example, the surnames "Aiauao" and "Chuah" have Soundex codes of "A000" and "C000" respectively. This problem needs to be addressed, as there are a significant percentage of Polynesian and Asian surnames, as well as other ethnic groups, in the Paxus database.

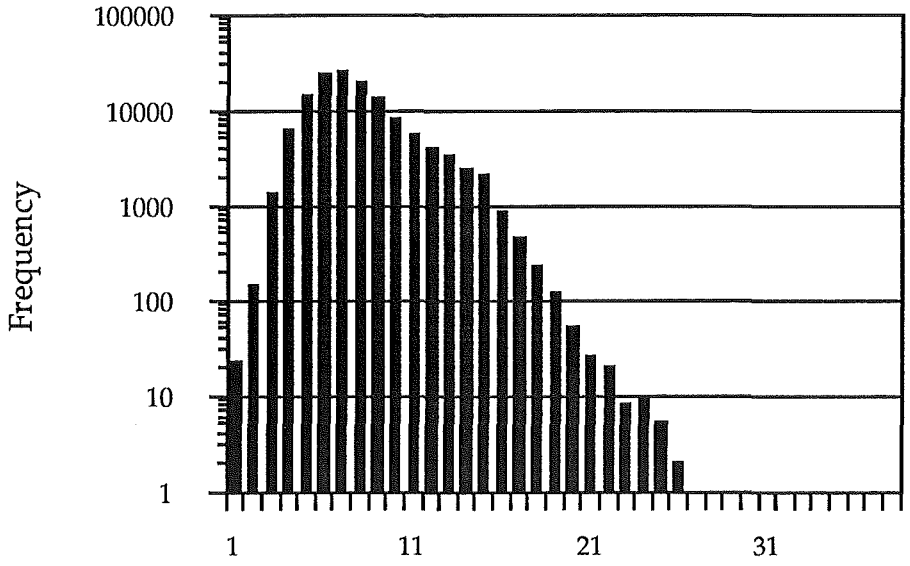


Figure 2.
Distribution of lengths of surnames in the database.

The surnames that will be returned in a given class are fixed, and there is no distinction within each class, and they are all judged equal. This is could be considered a major problem in a Dutch database, for example, as all surnames beginning with Van der will be grouped into one class, and the number of entries in such a class will be very high.

There are several avenues for improving the Soundex algorithm. One possiblity is lengthening the code to take account of longer surnames is one possibility. This gives more divisions in the classes with many distinct surnames, such as the "Van der" group. However, the negative effect of this strategy is that small classes are also divided, into even smaller classes, with many containing only 1 surname from the database, and this is almost as undesirable as the large groups with many distinct surnames, because these are no other possibly matching surnames within the class.

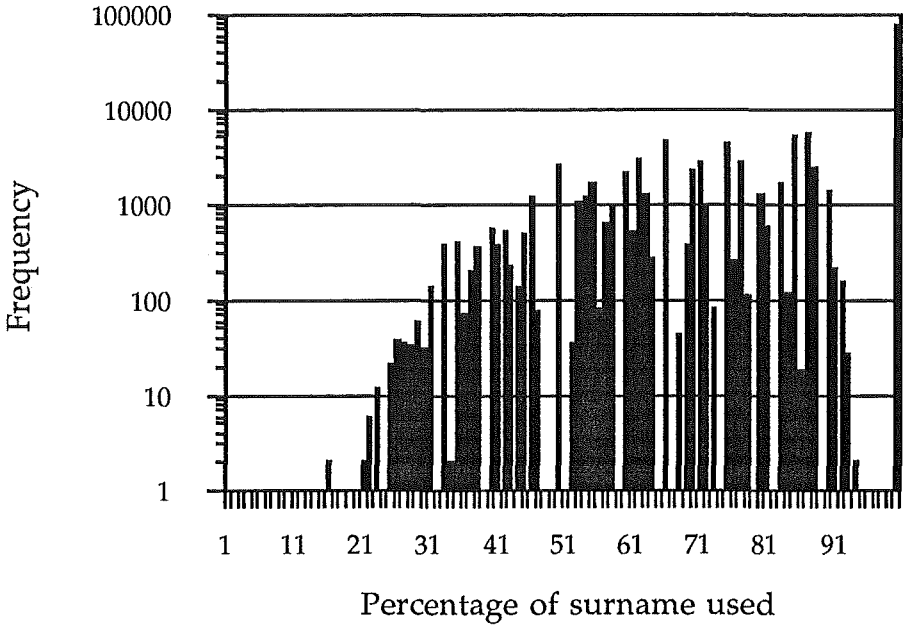


Figure 3.
Percentage use of each surname by Soundex algorithm.

Cutting back the generalization by retaining the vowels, 'H', 'Y' and 'W' is another possible improvement. While it achieves better results on surnames with a high proportion of vowels, the length of the codes must be increased to cover the same proportion of the surname. Because this has cut back the amount of generalization, it has effectively increased the range of classes, as in the previous improvement, and suffers similarly from being too specific.

Preprocessing the input is another available improvement. This has not been investigated in this study, but would take the form of converting groups of characters in the surname into smaller groups of characters. For example, the translation of "ph" to "f" would be used to capture variations in spelling of the same phonetic sound, although this would need to be carefully chosen to avoid generalizing other cases incorrectly. For example, the words "loophole", "haphazard" and "uphold" are cases where this translation should be applied.

3. Alternative methods for matching

The Soundex method is a classification type of algorithm, and performs poorly on a large variety of surnames distributed from different ethnic origins. Classification methods as a whole require careful initial design to ensure that the information sought is captured appropriately within few classes, and that this classification scheme will be robust even after additions of new surnames from distributions not yet encountered.

Approximate matching methods based on a distance of similarity suffer less from this effect than Soundex, in varying degrees depending on the algorithm. This type of algorithm has been chosen for investigation as an alternative to Soundex, although further research may produce a hybrid algorithm combing the relative speed of classes with the quality of results of similarity measures.

3.1 An Information Theoretic Approach

This method, developed by Bickel (1987), introduces a statistical basis for measuring the similarity bewtween two surnames. Before the algorithm is discussed, a short explanation of *information content* is needed.

The *information content* of a letter is a measure of the probability of that letter occurring in a surname. This value is termed the *entropy* of the letter, and is calculated using the following formula:

entropy of a character =
$$-\log_2 [\text{ proportion of character in the database }] \quad \text{(bits)}$$

Each different letter used in any of the surnames from the database is given a weight, related to the proportion of times it occurs in the database, and therefore the probability that it will be the next letter observed. Those that occur often are given a low weight, and a high weight is given to those that occur rarely.

The information theoretic approach determines a measure of distance between two surnames by summing the weights of their common characters. For example, with the alphabet "A", "B" and "C", the word "AC" is judged more similar to "BCA" than "A" (figure 4). Bickel calls this the *likeness* value of the two surnames. If this value is high, the surnames are judged good matches, otherwise they are judged poor matches. In the case of the Paxus database, the character set used allows whole number likeness levels ranging from 0 to 178, although not all whole numbers in this range are used, as the scale is not continuous.

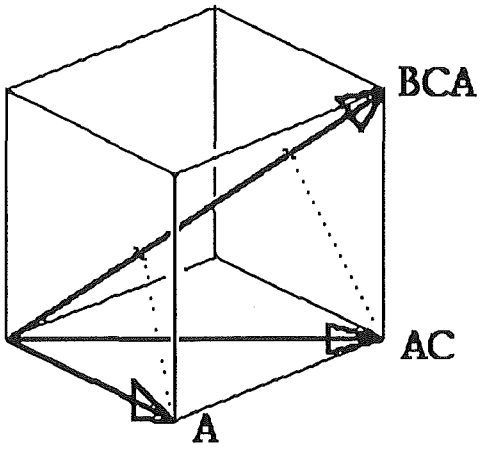


Figure 4
Information theory example

This method is useful for ignoring transposition of letters, which are generally caused by typographical errors. In fact, any two surnames that contain the same letters in any order (i.e. anagrams of each other) will generate a high similarity rating.

This approach has some problems associated with it. The Paxus database contains about 130 thousand distinct surnames, so the average number of different surnames per likeness value is about 700. In practice however, short names do not have many different letters that could be matched with, and consequently they cannot generate a large similarity value. As an example, "Smith" compared with "Smith" (i.e. an exact match), gives a likeness value of 24. This is the highest value that any surname matched with "Smith" can take. The range of likeness values actually used is much smaller than 178. The spread of maximum similarity values possible for each surname in the database is shown in figure 5.

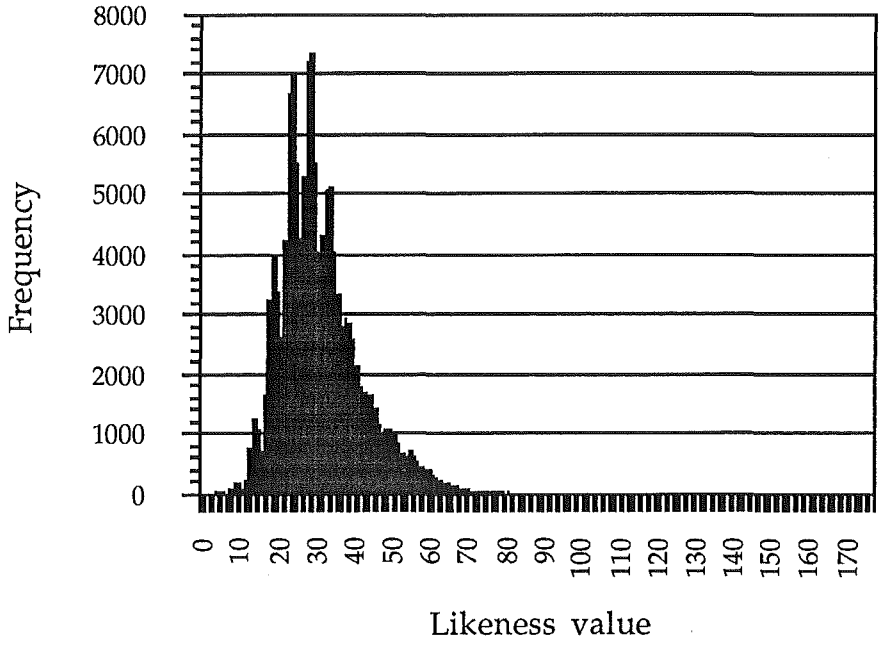


Figure 5
Spread of maximum "likeness" values.

This means that the 130 thousand distinct surnames in the Paxus are actually spread over a much narrower band than 0 to 178, and there will be many more names in some levels than the average of 700. Figure 6 shows the spread of likeness values when a search for the surname "Smith" is performed.

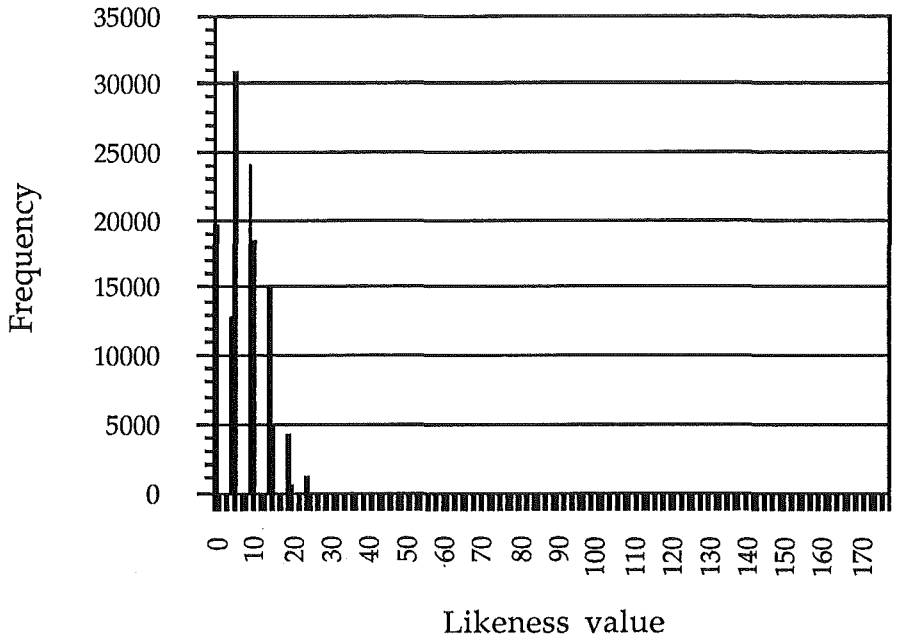


Figure 6
Spread of "likeness" values for a search on "Smith".

The example search above for "Smith" shows that there are very few likeness values actually used in a typical search. It also shows that the number of surnames in for each used likeness value is large returned by such a search is large. A related problem to the above is the case of long surnames. Having more letter positions available gives the possibility of a wider spread of letters from the alphabet, i.e. long surnames have a superset of the letters found in short surnames, and a search for a short surname returns many longer surnames, some of which bear little resemblance to it. These 'false matches' could be filtered out by some other measure of similarity, but has not been tested in this study as other methods were judged to be more effective.

Another problem is that this algorithm ignores the position of a letter in surnames. It is useful on a local scale for errors such as the transposition of two adjacent characters, but completely ignores phonetics on global matching of two surnames. For example the surnames "Stephen" and "Nespeth" will give the same likeness value when matched with "Stephens", although the second is not a likely candidate.

Many of the similarity values used by this approach can be arrived at by different common characters between surnames. For example, the likeness value of 22 can be generated by summing the weights 4, 5, 6 and 7, and also by summing the weights 4, 8 and 10. As a result of this effect, there are a large number of unrelated surnames in many of the simillarity ranges used by the algorithm.

3.2 The 'Edit Distance' Approach

The concept of edit distance was developed by Levenshtein and Damerau (Levenshtein, 1966; Wu and Manber, 1991; Faloutsos, 1985), and is regarded as the most common measure of closeness between two words (Wu and Manber, 1991). It is based on 4 different typographical errors:

- insertion of a character into a word,
- deletion of a character from a word,
- substitution of one character for another,
- transposition of two adjacent characters.

For example, the surnames "Smith" and "Amith" are judged to have an edit distance of 1, as 1 editing operation of the type mentioned above is needed to transform the first name to the second.

Typographical errors such as the four above are estimated to make up 80 percent of errors in words (Faloutsos, 1985; Peterson, 1980), and a searching method based on such errors should therefore match a large proportion of mis-typed words. This can also be used with similar surnames that are not typing errors, such as "Smith" and "Smythe".

Matching surnames using this method gives a natural measure of similarity, ranging from 0, where a surname is matched against itself. When performing a search, the algorithm is presented with the entire database, and will produce a list of all the surnames which are within a threshold edit distance from the surname used in the query. If this threshold level is large, it will allow many edit operations, and many surnames will be returned. For example, if a search for the surname "Smith" is entered into the Paxus database, allowing up to 3 edit operations, surnames including "Dit" and "Naysmith" are returned, if they exist.

This method gives more phonetically appropriate results than Soundex, because differences of single letters do not create a very large difference in phonetic sounding of the surname. For example, the Paxus database contains the surnames "Smith" and "Amith". The surname "Amith" is an entry in the database that probably contains an error; 'S' and 'A' are adjacent on QWERTY keyboards. Although both surnames are phonetically similar, Soundex will generate keys S530 and A530 respectively, which it would not class as a match, whereas the edit distance metric for these two surnames is 1, showing them to be typographically very close.

The implementation of an exact edit distance measuring algorithm is simple, but is intensive on computing resources. After a query is entered, the database is searched for all permutations of the query surname, within a specified number of edit operations. A heuristic form of the metric has been developed, that finds subsets of patterns differing from the query pattern by a specified number of edit operations (Wu & Manber, 1991). This can be implemented very simply in low

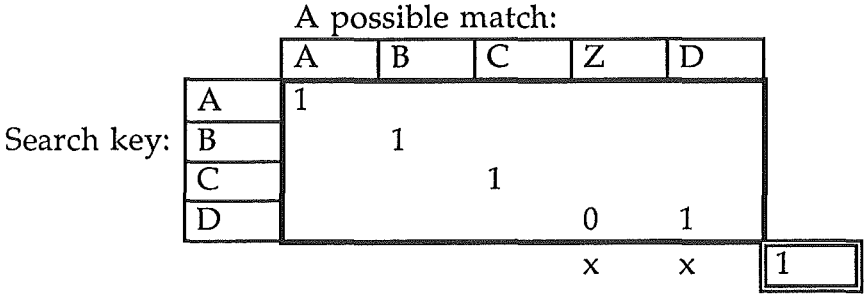
level code or in a hardware device to give a very fast matching device. I have modified this heuristic to apply to entire surnames only, rather than parts of the surname that are within the allowed number of edit operations.

The heuristic calculates an edit distance measure by moving a flag, which represents the state of the match so far, through a 2-dimensional table. If there is a flag in the table, in a position where the characters at the end of the same row and column match, then the flag moves diagonally down to the right by one step. If not, the flag is split into 3 new flags, and they move directly down, diagonally down and right, and right by one step. Each of these splits is counted as an edit operation. A match is signalled if there is a '1' in the highlighted position on the lower right, and the number of edit changes done so far is within the allowed limits specified by the operator.

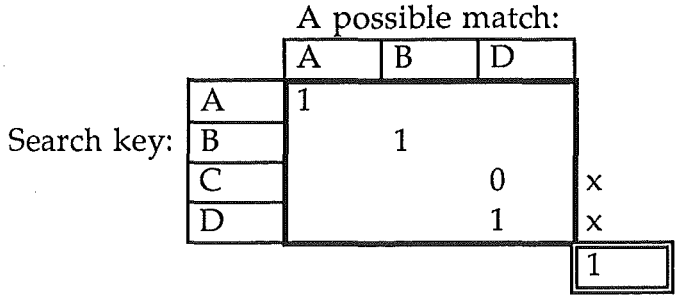
In the case of an insertion of a character, the character in the column to the right of the flag should match the character in the flag's row. If there is a deletion of a character, then the character on the row below the flag should be equal to the character at the top of the flag's column (figure 7b). For substitution errors, the current character is ignored, and the flag moved diagonally down and right as in the case for a normal match.

Some examples of a search for the pattern "ABCD" are given in figures 7(a), 7(b) and 7(c).

7(a). Insertion of a character:



7(b). Deletion of a character:



7(c). Substitution of a character:

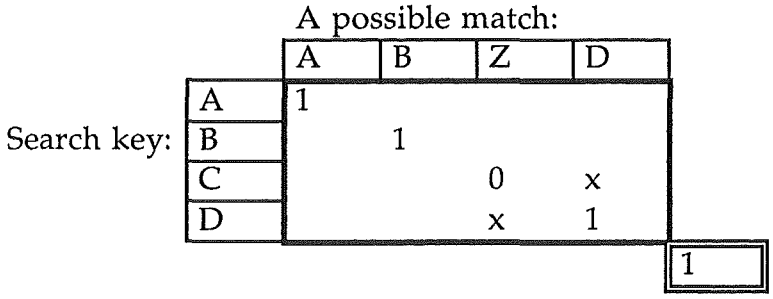


Figure 7
Examples of the fast edit distance heuristic

This method is not exact in some cases, and transpositions are counted as two editing operations, but the cases where these are important are significantly few to be ignored. For example, "Sirith" is a surname contained in the Paxus database. The heuristic claculates an edit distance of 3 between this surname and the surname "Smith" (deletion of the 'M' then insertion of 'R' and 'I'), although the actual edit distance between these two surnames is 2 (substitution of an 'I' for the 'M', and insertion of an 'R'). As mentioned before however, the number of edits allowed should be kept low for short names, but the Paxus database has names up to 28 characters in length, and edit distance is not very effective in retrieving these, as the number of differing characters can be higher. For example, a search for the surname "Van der Poel" allowing 4 edits, returns over 200 surnames, but reasonable results are obtained from a search for this surname allowing only 2 edit operations. Such effects limit the usefulness of the edit distance approach.

3.3 The Common N-grams method

This approach is similar to the information theoretic approach discussed earlier, except that it uses groups of characters, instead of single letters for comparisons. It is discussed briefly in Faloutsos (1985). An N-gram is a set of N consecutive characters from a surname. For example, the digrams (pairs of consecutive characters) in the surname "Hamilton" are (the method is case insensitive) "•H", "HA", "AM", "MI", "IL", "LT", "TO" and "ON".

For the first character in a surname, the special symbol "•" is used to denote the beginning of the surname. If N is more than 2, this character is added to the beginning of each N-gram at the beginning of the word. For example, if N was 3 in the example above, then the first few N-grams would be "••H", "•HA", and "HAM".

The common N-grams method of approximate matching builds a table of N-grams in the surname used in the query. Each surname used in the database is then searched to see how many of these N-grams it contains. An example of the table is given in table 3.

Searching for:	Possible matches:	
"Caltex"	"Carlton"	"Cotton"
•C	•C	•C
CA	CA	
AL		
LT	LT	
TE		
EX		

Table 3
Example of common N-gram method, with N=2.

When N=1 (i.e. only 1 character is put into the table at each step), this method is similar to the information theoretic approach.

Because this method uses N-sized groups instead of single characters, the results are of higher quality (i.e. more similar surnames are returned, and more dissimilar surnames are ignored, and not returned). This effect occurs partially because a group of consecutive characters resembles a syllable from a surname, whereas a single character does not capture that much information; i.e. position in the surname is implied, and consequently phonetically similar surnames are returned, and partially because positional information is caught in varying degrees by the N-grams.

For example, consider the di-grams "•A", "AB", "BC" and "CD". There is only one word that these di-grams could be generated from, namely "ABCD", and so the

This method is a distance method, because results of a search throughout the entire database can be sorted by the number of common N-grams that are found for each surname. Higher values numbers of common N-grams give a closer match, which contrasts with the edit distance approach above where higher numbers of allowed edits returns more distant surnames.

The common N-gram method does not work well on short surnames. If one character is different between two surnames, then up to 'N' N-grams are affected, and therefore different. Consequently, the number of different N-grams affected does not vary linearly with the number of different characters (figure 8). Also, a short surname does not conatin many N-grams, so it is not easy to discriminate between other surnames. If a 3 character surname is used in a query with N=2, then there are only 4 N-grams that the surname can have in common with any

other surname, and many surnames may have at least one or two of these N-grams.

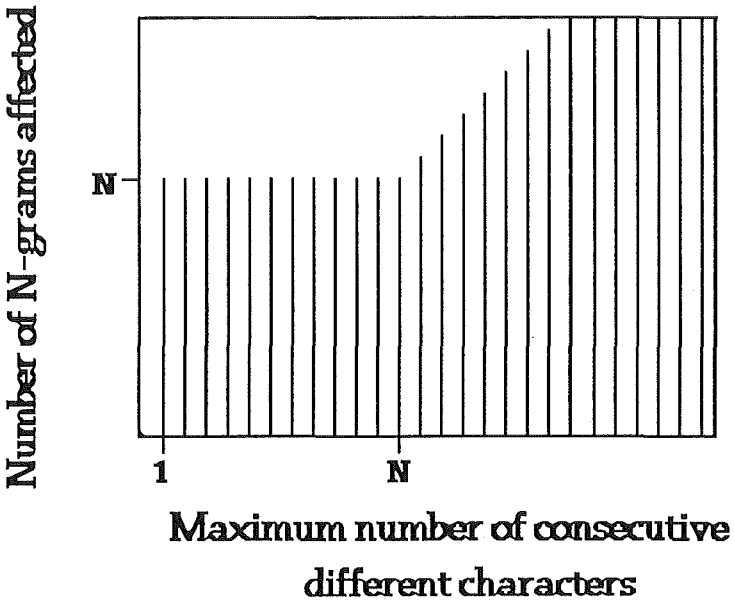


Figure 8
Relationship between minimum number of N-grams affected, and number of differing characters.

For example, if the surnames "Ait" and "Ayt" are compared with this method, with N=2 (i.e. digrams) then they will not be considered as good matches, as they have only 1 common N-gram: "•A", although they are both phonetically similar. This effect drops considerably as the length of surnames to be matched is increased, but because of the effect mentioned above, of up to N N-grams being affected with a 1-character difference, N should be chosen to be less than half the average length of surnames in the database. It is also related to the length of the surname that is being searched for.

The speed of this method could be improved, by storing the N-grams of each surname with the surname, in a hash table for example. This would mean that claculations would only be done once, and the algorithm would only need to perform comparisons for each surname.

3.4 The "Theta Proximity" algorithm

This algorithm, developed by Proximity Technology Ltd. (Rosenthal, 1984; Taylor, 1986) performs a similarity function by finding how many characters are in common positions in two names. It bases more emphasis on the startings and endings of words, rather than the middle, and transpositions detract less from the similarity of a word than deletions or insertions. The algorithm works

forward adding up the number of matched characters in successively longer passes over the surnames, then repeating the process in the reverse direction. An example of its operation in comparing "Smith" and "Smythe" is given in figure 9.

Forward direction:

Iteration	1	2		3			4				5				
Query name:	S	S	M	S	M	I	S	M	I	T	S	M	I	T	H
Possible match:	S	S	M	S	M	Y	S	M	Y	T	S	M	Y	T	H
Value of match:	1	2	3	4	5	5	6	7	7	8	9	10	10	11	12

Reverse direction:

Iteration	6	7		8			9				10				
Query name:	H	H	T	H	T	I	H	T	I	M	H	T	I	M	S
Possible match:	E	E	H	E	H	T	E	H	T	Y	E	H	T	Y	M
Value of match:	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12

Figure 9
Example of the "Theta Proximity" algorithm, on the surnames "Smith" and "Smythe".

The algorithm searches forward through two strings supplied to it, counting the number of matched symbols. It compares the first characters of each string, then the first two characters, then the first three, and continues in this way to the end of the shortest string. It then repeats the operation in the reverse direction, and sums the two results. The similarity of the two strings is measured by applying this value for each string in the function below:

$$\text{similarity} = \frac{2 * \text{match (string1,string2)}}{\text{match (string1,string1)} + \text{match (string2,string2)}}$$

For the example above, the similarity of the surnames "Smith" and "Smythe" is calculated to be 0.34. The algorithm then applies a threshold of typically 0.5 to the similarity values, discarding those surnames with similarity values less than this threshold. The example surnames given above would therefore not be considered as matches.

4. Evaluation

The aim of this project is to find a better method of approximate matching than Soundex, in terms of quality of results (i.e. the surnames returned from a search should be related closely to the surname searched for), and if possible, the time taken should be at most, as long as that taken by an equivalent search using Soundex. In grading an algorithm's performance for approximate matching, there is no best method that works for all surnames likely to be encountered. Therefore, the methods used to evaluate the algorithms discussed in this report will be based on the performance of the Soundex algorithm currently in use at Paxus Health.

To determine algorithm A's performance, in relation to algorithm B, results for algorithm B's searches are required for different types of queries. Where algorithm B performs poorly for a search, algorithm A should perform well, to justify using it in preference to B. If algorithm A is to be considered, it also needs to perform as well as algorithm B on searches where B performs well. When the Soundex algorithm is used by Paxus Health for searching surnames in their database, certain surnames tend to give poor results. For example, when searching for surnames with the "Van der" prefix, the Soundex algorithm will return over 900 surnames from the Paxus database. This is judged too high, and therefore the number of different surnames per random query is regarded as the basis for comparison between each method.

A measure of the Soundex algorithm's perfomance can be gained from determining how much information Soundex is using to construct a code (figure 3). Subjective measurements have also been studied with the surnames that each method returned for a given query, as these surnames may contain little resemblance to the surname involved in querying the database.

Ideally, surnames in the database should be widely spread out on a scale of similarity; i.e. an algorithm needs to be found where a small difference in two surnames gives a large difference in the similarity rating of the two names. The concept behind this is that it will allow much more control over the surnames retrieved from a query. As an example of an algorithm that does not exhibit this feature, the Information - theoretic approach mentioned above uses only a small proportion of its predefined similarity values for most surnames likely to be encountered at Paxus (figure 5), whereas the edit-distance metric discussed earlier provides roughly the same range of similarity values, but the distribution of surnames is sharply skewed, giving a high degree of control over the range of no edit operations to 2 or 3 edit operations. Consequently, the edit - distance metric is preferred over the Information Theoretic approach as a method of approximate matching.

A series of graphs follows, presenting the spread of surnames returned from searches on the database. These are to show the relationship between the number of different unique surnames returned for a search, and the total number of

records that would be returned from the main database with a search for this surname using the algorithm discussed. The main aim of the project in reducing the number of different surnames retrieved for each search, or at least limiting it to those that are relevant, involves determining which algorithms can produce a lower y-axis scale.

Reducing the total number of surnames returned from a search can be done by searching on other fields as well as surname, but to reduce the number of different surnames returned requires a good replacement for the Soundex algorithm.

The Soundex algorithm:

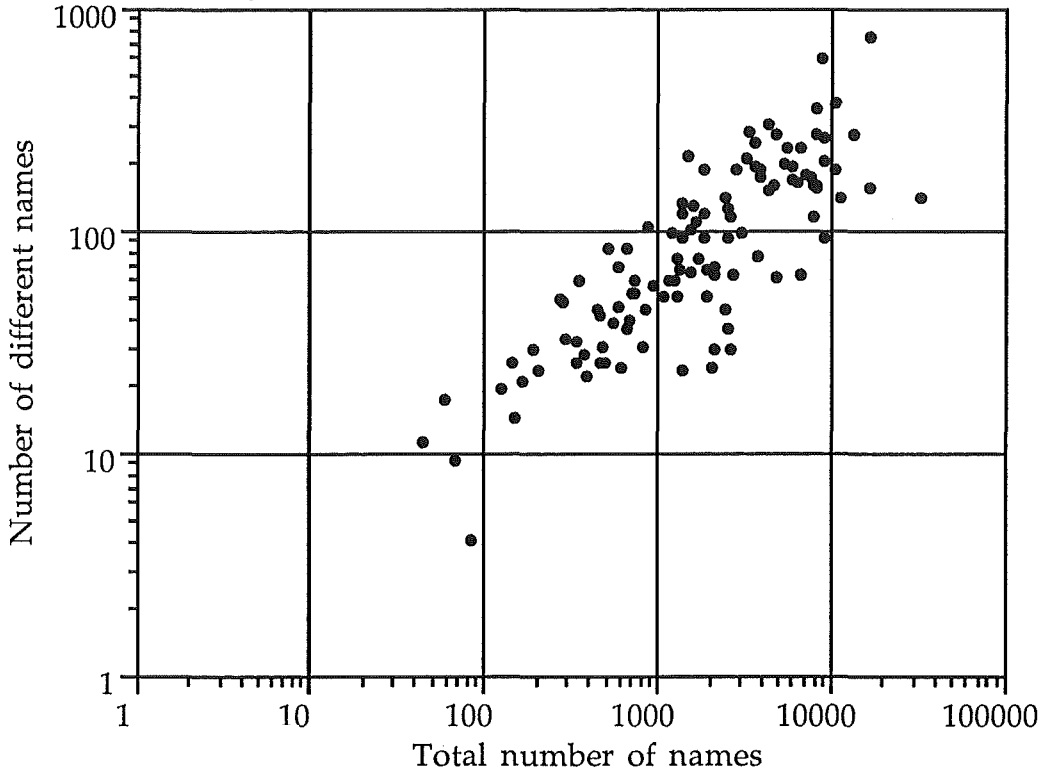


Figure 10(a).
Spread of search results for the Soundex algorithm,
(Soundex average search conditions)

Figure 10(a) shows the average spread of names returned from a range of queries, using the Soundex algorithm. As can be seen from the graph, there are some searches that return a large number of different names. This is the main problem with the Soundex algorithm that Paxus Health would like to reduce.

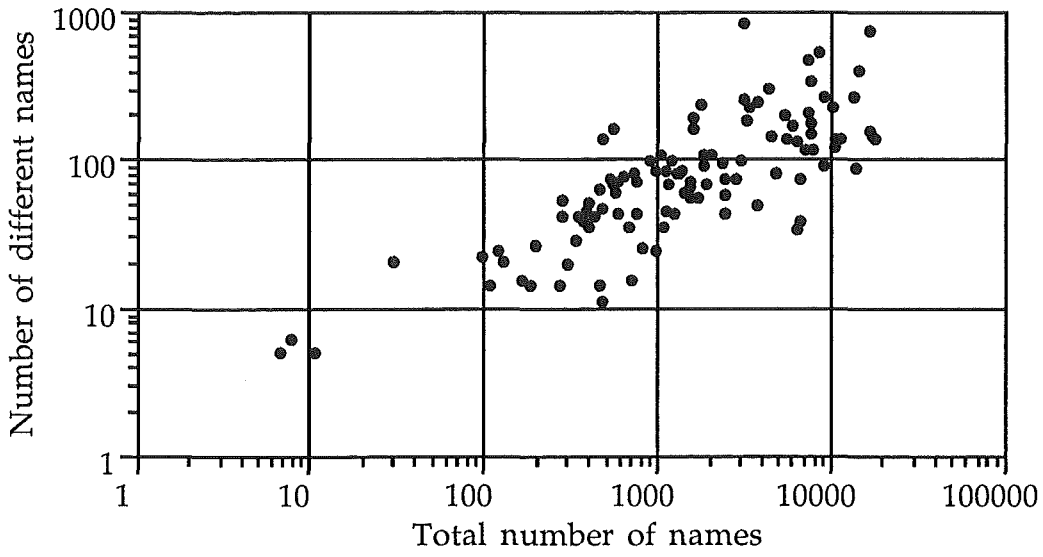


Figure 10(b)
Spread of search results for the Soundex algorithm, (Soundex poor search conditions)

More noticeably with poor search conditions, Soundex returns a large number of different names per search (figure 10b).

The edit distance hueristic approach:

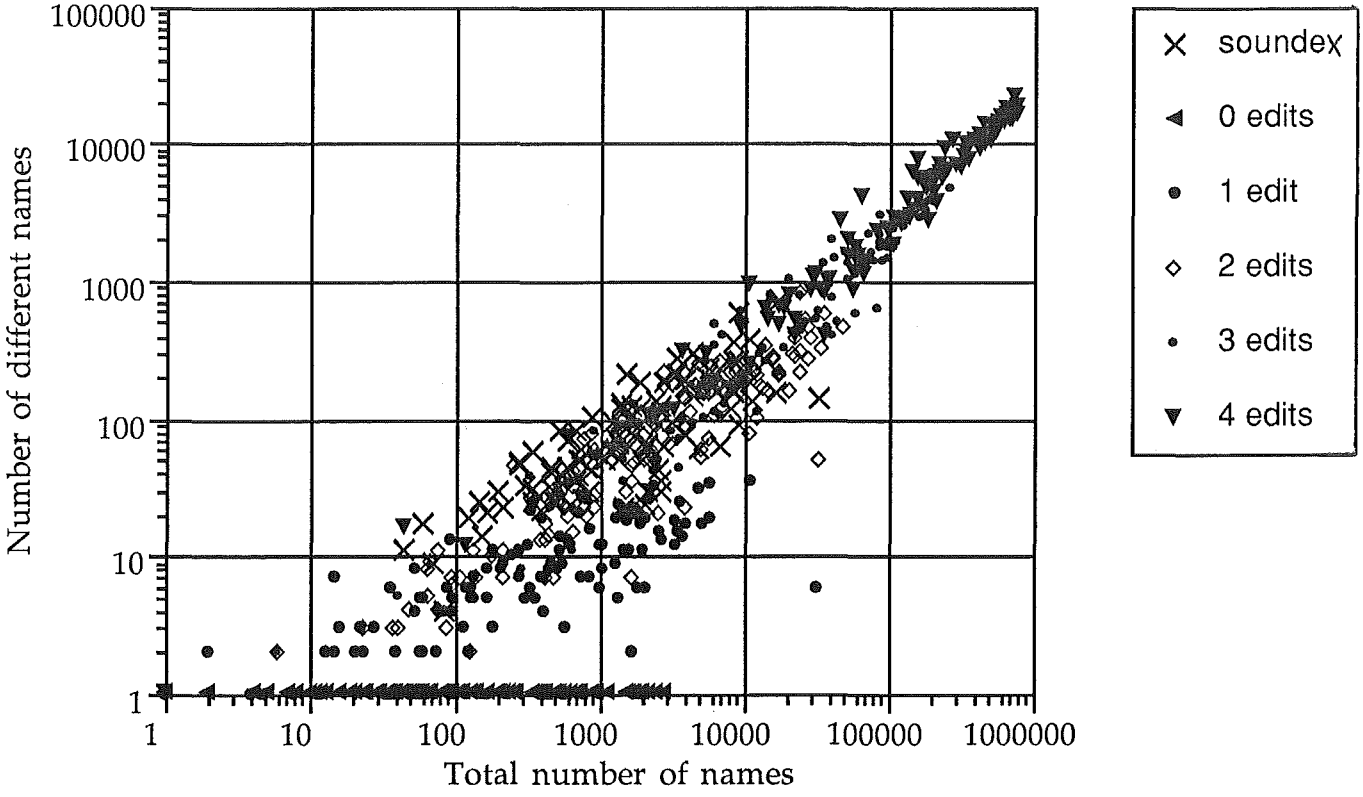


Figure 11(a)
Spread of search results for the edit distance heuristic (Soundex average search conditions)

Figure 11(a) shows the results of a search using the edit distance heuristic. This graph shows that the number of edit operations should be restricted to 1 or 2, to give better performance than Soundex. It also shows that the results of a search can be varied, between a small amount and a large number of surnames being retrieved.

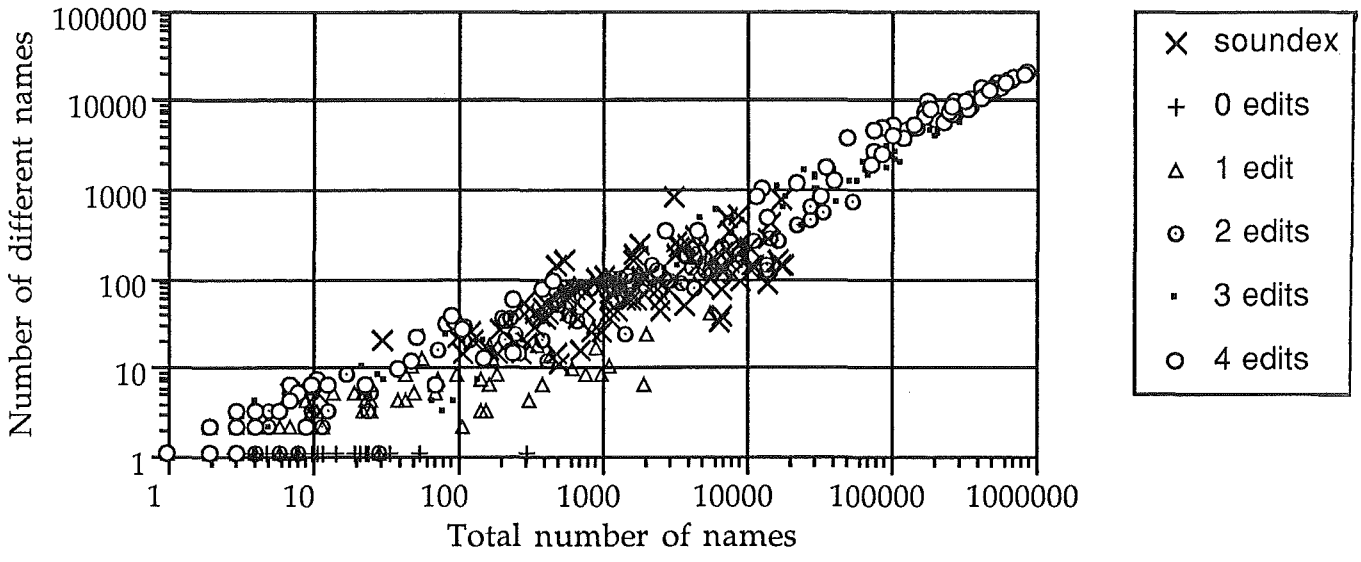


Figure 11(b)
Spread of search results for the edit distance heuristic (Soundex poor search conditions)

Figure 11(b) shows the spread of results for a case where Soundex performs poorly. As with the previous edit distance metric results, the range of surnames returned with 1 or 2 edit operations allowed, is still significantly less than Soundex on many cases.

The common N-grams approach:

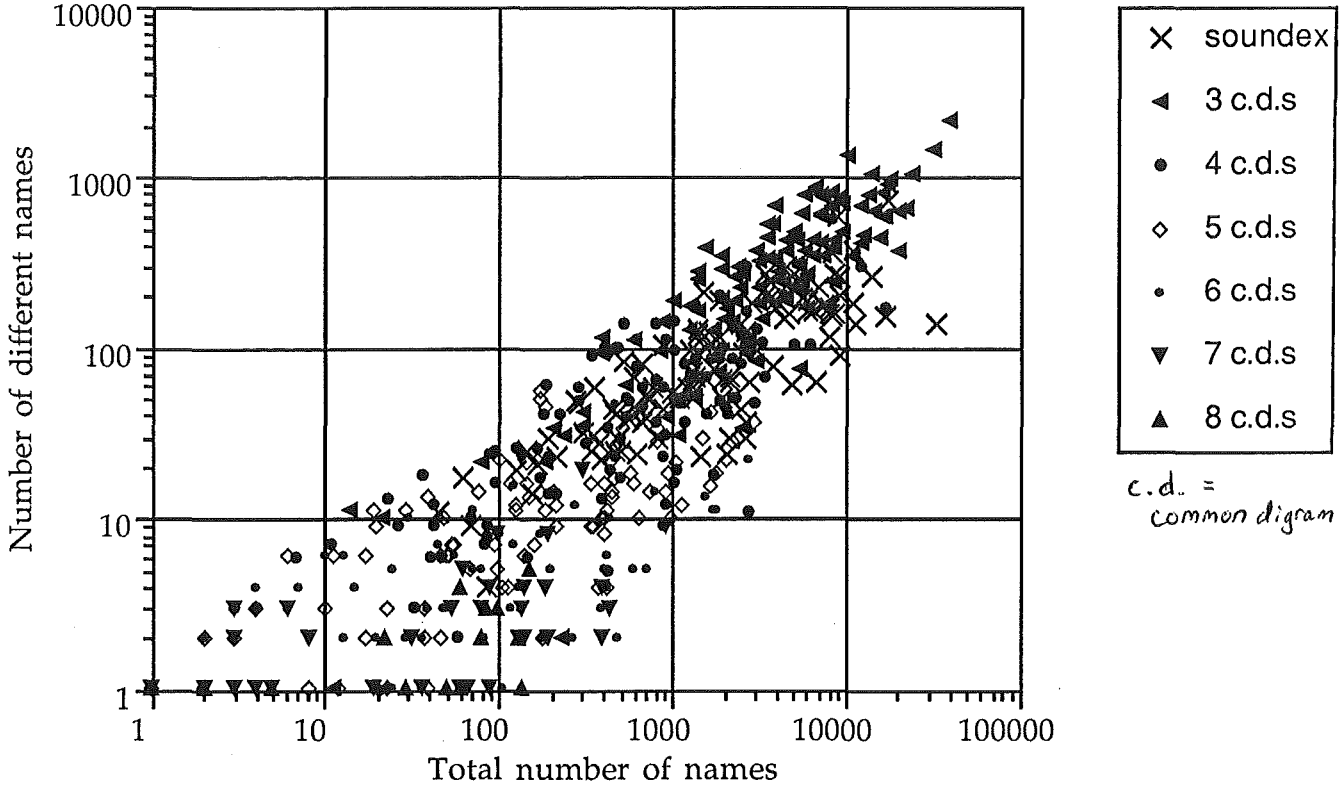


Figure 12(a)
Spread of search results for the common N-grams algorithm, N=2 (Soundex average search conditions)

As with the edit distance heuristic, the common N-grams approach provides a variable range of results for a sesarch. For the Paxus database with N equal to 2, choosing 4 or 5 common N-grams as a minimum level would produce acceptable results in terms of the number of surnames returned.

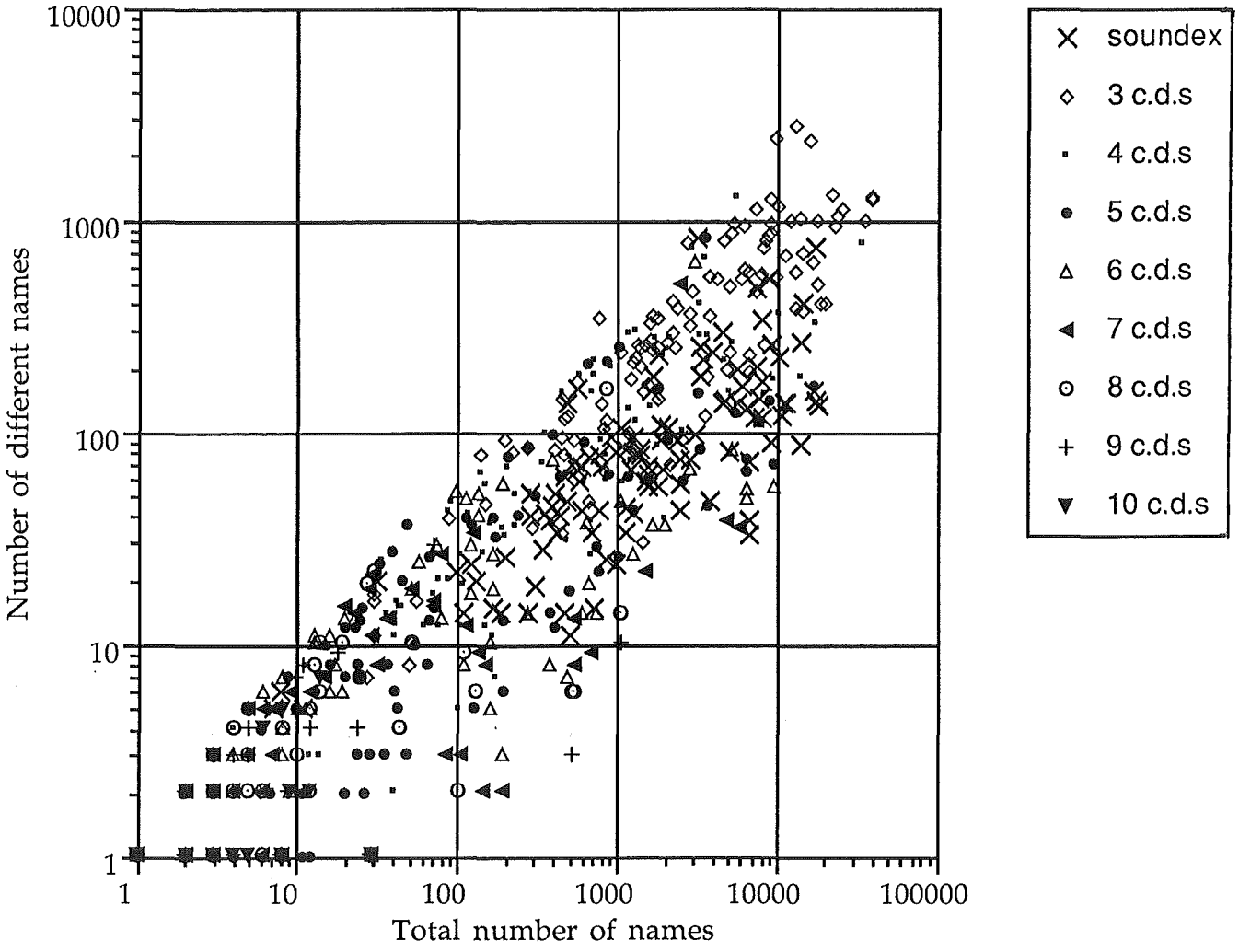


Figure 12(b)
Spread of search results for the common N-grams algorithm; N=2 (Soundex poor search conditions)

In figure 12(b), the results are similar for those of the edit distance approach. With N=2, at least 4 or 5 common N-grams will give better results than Soundex, in terms of the number of surnames retrieved.

The Theta Proximity algorithm:

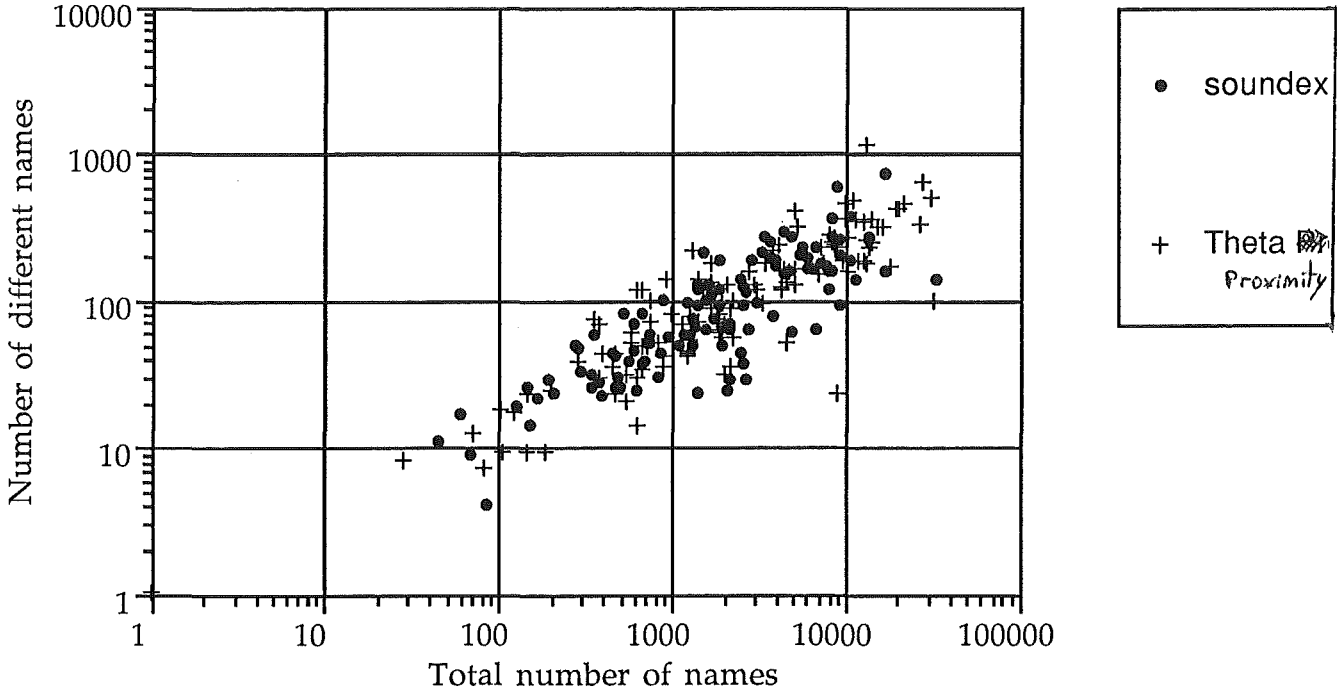


Figure 13(a)
Spread of search results for the Theta Proximity algorithm, (Soundex average search conditions)

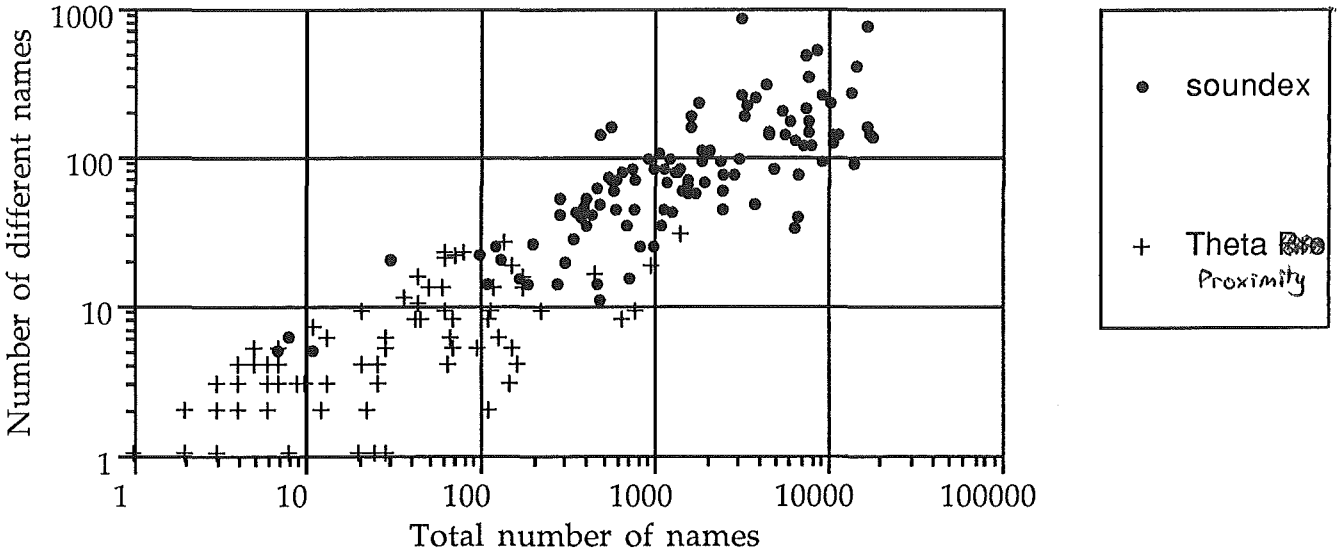


Figure 13(b)
Spread of search results for the Theta Proximity algorithm, (Soundex poor search conditions)

The spread of results for the Theta Proximity algorithm is roughly equivalent to that of Soundex for the average case, and a lot lower for the case where Soundex performs badly. However, when observed closely, the surnames returned by this

method failed to include many subjectively probable matches, that are returned in results from the other approaches.

Subjective measurements:

On examination of the ouputs of each method, the edit distance heuristic provided the most appropriate results to the query that was entered, and all except the information - theoretic approach produced superior search results to that of Soundex.

5. Conclusions

All of the algorithms discussed above, with the exception of the Information Theoretic approach, have performed as well as, if not better than, the Soundex algorithm. The edit distance heuristic and common N-grams method have the best performance out of all the algorithms studied and should be considered seriously as replacements for the Soundex algorithm in current use at Paxus. The edit distance heuristic gave better results than the common N-grams approach, and with the exception of the information theoretic approach, all the algorithms were superior to the Soundex algorithm.

The Information Theoretic approach was judged as giving poor results in comparison with the other alternatives, although this method would be acceptable in different circumstances, such as a much smaller variety of surnames. However, given the relative ease of coding a fast edit distance heuristic, the information theoretic measure does not need to be used.

The Theta Proximity algorithm does not capture the kind of differences between words that are desired, such as phonetic information. It relies too heavily on each surname being roughly the same length, and does not handle insertions or deletions very well in that they can drastically affect the similarity rating of two surnames. This is because they cause one surname to become offset from another by shifting he characters in it to the left or right, which makes character by character comparison useless. However, substitutions and transpositions are handled very well by the method, since they do not affect the length of a surname, and consequently do not offset the surname.

Future research is possible in a number of areas. The adoption of a hybrid algorithm that combines the benefits of the common N-grams approach for long surnames, and edit distance for shorter surnames should be studied, as the edit distance method alone is constrained by the time delay of searching every surname in the database. Variations in N for the common N-grams method could also be studied, as the algorithm was only tested with N=2.

Implementation of the algorithms could be studied in depth, as there are many possible methods of implementing the algorithms. For example, some investment in special-purpose hardware for text matching may be appropriate (the Theta Proximity algorithm is implemented on a commercially available integrated circuit), as this would give high speed matching capabilities to the database. Also the actual algorithm implementation itself is open for experimentation. Efficient methods for the implementation of the chosen algorithm(s) should be studied.

More knowledge of the database itself would be very useful in determining a good algorithm to use, and setting its parameters accordingly; i.e. tailoring the chosen algorith(s) to fit the data for Paxus Health should give good improvements in the effectiveness of the searching operation.

To summarize, the problem of approximate matching has been presented. Issues that arise and need to be considered have been discussed, and a number of algorithms for partial solution of the problem have been displayed. A recommendation is made, and futre research work outlined.

6. References

Baeza - Yates, R. (1989) "Algorithms for string searching: A survey", SIGIR, 23, #3, 4.

Bickel, M.A. (1987) "Automatic correction to unspelled names: a fourth generation approach", CACM, 30, #3, March.

Faloutsos, C. (1985) "Access methods for text", Computing Surveys, 17, #1, March.

Knuth, D.E. (1973) "The Art of Computer Programming", Addison - Wesley.

Levenshtein, V.I. (1966) "Binary codes capable correcting deletions, insertions, and reversals", Sov. Phys. Dokl., February, 707-710.

Peterson, J.L. (1980) "Computer programs for detecting and correcting Spelling Errors", CACM Dec. 1980, 23, #12.

Rosenthal, S. (1984) "The PF474: A coprocessor for string comparison", BYTE, November.

Taylor, D. (1986) "Wordz that almost match", Computer Language, November.

Wu, S. & Manber, U. (1991) "Fast text searching with errors", Technical report TR 91-11, Department of Computer Science, University of Arizona.